



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
-----------------	-------------	----------------------	---------------------	------------------

10/816,052

03/31/2004

Thomas A. Piazza

42P19136

9615

45209

7590

08/17/2009

INTEL/BSTZ

BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP

1279 OAKMEAD PARKWAY

SUNNYVALE, CA 94085-4040

EXAMINER

CRAWFORD, JACINTA M

ART UNIT

PAPER NUMBER

2628

MAIL DATE

DELIVERY MODE

08/17/2009

PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No. 10/816,052	Applicant(s) PIAZZA ET AL.	
	Examiner JACINTA CRAWFORD	Art Unit 2628	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 04 May 2009.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-4,6-14,16-24 and 27 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-4,6-14,16-24 and 27 is/are rejected.
- 7) ☐ Claim(s) _____ is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413) |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | Paper No(s)/Mail Date. _____ |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08) | 5) <input type="checkbox"/> Notice of Informal Patent Application |
| Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

Claim Objections

1. Claim 27 is objected to because of the following informalities: Claim 27 is dependent upon cancelled claim 26. Appropriate correction is required.

However, claim 27 is believed to be dependent upon independent claim 23 and is rejected accordingly.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

3. Claims 1-4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Andrews et al. (US 2005/0122339) in view of Rosenbluth et al. (US 2003/0046488).

Art Unit: 2628

As to claim 1, Andrews et al. disclose a graphics processor, comprising:
a multithreading, multi-core graphics engine to process pixel data (Figure 1, 102: note CPU1, CPU2, ...CPU_n denotes multi-core; Figure 2 shows threads; [0033] notes processing “geometry data” which can be considered “pixel data”);
a render-cache, readily accessible to the graphics engine, to store pixel data (Figure 1, shared L2, 122 accessible to processor 102 and each processor including its own internal cache; [0042] and [0043]); and a render cache controller to maintain data coherency between the render-cache and a main memory ([0050] notes coherency module).

Andrews et al. disclose the threads comprises series of subtasks performed in a specific order forming a sequence of such subtasks and the tasks must be executed in the order for rendering scenes [0054] and pixel data specifically stored in cache-line address and if information is valid or invalid [0064].

However, Andrews et al. do not expressly disclose maintaining the order in which each thread is dispatched to the graphics engine in line with the multithreading, multi-core graphics engine processing the pixel data corresponding to each thread and wherein the render-cache controller is to block a thread from dispatching to the graphics engine if the thread specifies a cache-line address of the render-cache containing a pixel in flight.

Art Unit: 2628

Rosenbluth et al. disclose a controller to maintain the order in which each thread is dispatched and executed in line with the multi-threading, multi-core processing the data corresponding to each thread ([0074], [0075], [0083] thru [0086] notes that the threads are executed in a strict sequential order and a thread is not executed until the previous is complete; CAM is used to maintain coherency and ensure correct sequential processing) and wherein the render-cache controller is to block a thread from dispatching to the graphics engine if the thread specifies a cache-line address of the render-cache containing a pixel in flight ([0084] thru [0086] notes look-up operation and results of cache hit/miss and "in-flight" status).

Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to modify Andrews et al. system with Rosenbluth's method of maintaining order of threads to ensure the execution of threads in a proper order which will accurately render scenes to be displayed.

As to claim 2, Andrews et al. modified with Rosenbluth et al. disclose the graphics processor including:

raster logic (data generating logic) to generate threads, each thread including at least one cache-line address indicating the location of the pixel data in the render-cache (Andrews, Figure 3, 302, 304; [0059], lines 1-18; [0064] notes that the data address corresponding to the cache-line address); and a thread

Art Unit: 2628

dispatcher to dispatch each thread to the graphics engine only when the render-cache controller indicates that the at least one cache-line address is valid (Andrews, Figure 4 notes validity; Figure 5 notes dispatching threads; [0064]).

As to claim 3, Andrews et al. disclose the graphics processor wherein the multithreading, multi-core graphics engine is to process pixel data for rendering 3D graphics [0033].

As to claim 4, Andrews et al. modified with Rosenbluth et al. disclose wherein at least one cache –line address is valid if the render-cache controller indicates a cache hit during a look-up operation, and the pixel data stored at the at least one cache line address is not in flight (Andrews, [0064] notes pixel data specifically stored in cache-line address and if information is valid or invalid; Rosenbluth, [0084] thru [0086] notes look-up operation and results of cache hit/miss and "in-flight" status).

4. Claims 6-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Andrews et al. (US 2005/0122339) in view of Rosenbluth et al. (US 2003/0046488) as applied to claim 2 above, and further in view of Hussain (US 2004/0233208) and Chrysos et al. (US 6,549,930).

As to claim 6, Andrews et al. modified with Rosenbluth et al. disclose a cache system (Andrews, Figure 1, shared L2, 122) where cache-lines have different fields to indicate certain information about the cache and the data such as cache-line address (Andrews, [0064]) and indicating whether data is “in-flight” (Rosenbluth, [0086]), but do not disclose the graphics processor wherein the render-cache controller comprises: a content addressable memory to map pixel coordinates to a cache-line address of the render-cache, the address specifying a location in the render-cache where pixel data corresponding to the pixel coordinates is stored;

a pixel mask array having a bit for every entry of the content addressable memory, each bit to indicate whether previously allocated pixel data is in flight; and, a cache-line status array with a bit for every entry of the content addressable memory, each bit to indicate the availability of a cache-line in the render-cache.

Hussain discloses the graphics processor wherein the render-cache controller comprises: a content addressable memory (tag compare unit) to map pixel coordinates to a cache-line address of the render-cache [0043], the address specifying a location in the render-cache where pixel data corresponding to the pixel coordinates is stored [0035](it is obvious that addresses are used to specify a particular location in memory for data and information to be stored

Art Unit: 2628

and retrieved); and a cache-line status array (cache) with a status bit for every entry of the content addressable memory [0032], each status bit to indicate the availability of a cache-line in the render-cache ([0044] thru [0046] notes that the state per cache line within the tag compare unit indicates availability).

Chrysos et al. disclose a data array having a bit for every entry of a content addressable memory, each bit to indicate whether previously allocated data is in flight (column 14, line 54 thru column 15, line 3 notes a sample bit used to indicate a in-flight instruction).

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Andrews et al.'s cache system with Hussain and Chrysos et al.'s cache system since using bits common way of allocating data that is read and written to cache which maintains coherency of the memory for future processing.

As to claim 7, Andrews et al. modified with Rosenbluth et al., Hussain, and Chrysos et al. disclose the graphics processor wherein the render-cache controller further comprises a pipeline interface to receive cache-line addresses when the graphics engine reads from or writes to the render-cache (Andrews, [0064]; Hussain [0047] thru [0050]).

As to claim 8, Andrews et al. modified with Rosenbluth et al., Hussain, and Chrysos et al. disclose the graphics processor wherein the render-cache controller maintains the in-flight status of pixel data stored in the render-cache by receiving the cache-line addresses from the pipeline interface (Andrews, [0064]; Rosenbluth, [0086] notes CAM to determine “in-flight” status; Chrysos, column 14, lines 53 thru column 15, line 3 notes bit stays asserted as long as the data is in-flight).

As to claim 9, Andrews et al. modified with Rosenbluth et al., Hussain, and Chrysos et al. disclose the graphics processor wherein the render-cache controller changes the status of pixel data stored at a particular cache-line address to indicate that the pixel data is in-flight when the render-cache controller receives the address of the cache-line via the pipeline when the graphics engine reads the pixel data from the cache-line associated with the cache-line address (Andrews, [0064]; Hussain, [0043] thru [0046]; Rosenbluth, [0085] and [0086]; Chrysos, column 10, lines 14-67; column 14, lines 53 thru column 15, line 3).

As to claim 10, Andrews et al. modified with Rosenbluth et al., Hussain, and Chrysos et al. disclose the graphics processor wherein the render-cache controller changes the status of pixel data stored at a particular cache-line

Art Unit: 2628

address to indicate that the pixel data is no longer in flight when the render-cache controller receives the address of the cache-line via the pipeline when the graphics engine writes the pixel data to the cache-line associated with the cache-line address (Andrews, [0064]; Rosenbluth, [0085] and [0086]; Chrysos, column 10, lines 14-67; column 14, lines 53 thru column 15, line 3 notes that the stays asserted only for the data that is in-flight).

5. Claims 11-13, 16-20, and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hussain (US 2004/0233208) in view of Chrysos et al. (US 6,549,930).

As to claim 11, Hussain disclose a render-cache controller comprising:
a content addressable memory (tag compare unit) to map pixel coordinates to a cache-line address of a render-cache [0043], the cache-line address specifying a location in the render-cache where pixel data corresponding to the pixel coordinates is stored [0035](it is obvious that addresses are used to specify a particular location in memory for data and information to be stored and retrieved); and a cache-line status array (cache) with a status bit for every entry of the content addressable memory [0032], each status bit to indicate the availability of a cache-line in the render-cache ([0044] thru [0046] notes that the state per cache line within the tag compare unit indicates availability).

Hussain differs from the invention defined in claim 11 in that Hussain does not disclose a pixel mask array having a mask bit for every entry of the content addressable memory, each mask bit to indicate whether previously allocated pixel data is in flight and the thread dispatcher is blocked from dispatching threads generated by raster logic if the threads include cache-line addresses of the render-cache containing the pixel data in flight.

Chrysos et al. disclose a data array having a bit for every entry of a content addressable memory, each bit to indicate whether previously allocated data is in flight (column 11, lines 64-67; column 14, line 54 thru column 15, line 3 notes a sample bit used to indicate a in-flight instruction) and the thread dispatcher is blocked from dispatching threads generated by raster logic if the threads include cache-line addresses of the render-cache containing the pixel data in flight (Chrysos, column 14, line 53 thru column 15, line 3 notes only one instruction is in-flight at one time until it retires or is aborted denoting other instruction are "blocked" from dispatching).

Chrysos et al. do not explicitly teach pixel data. However, Hussain discloses the pixel data and is used in view of Chrysos et al. system.

Art Unit: 2628

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hussain's system with Chrysos et al.'s cache system since using bits common way of allocating data that is read and written to cache which maintains coherency of the memory for future processing.

As to claim 12, Hussain et al. modified with Chrysos et al. disclose the render-cache controller comprising: a pipeline interface to receive a cache-line address when a graphics engine reads or writes pixel data to the render-cache (Hussain, [0043] thru [0050]).

As to claim 13, Hussain modified with Chrysos et al. disclose the render-cache controller wherein the pixel mask array is to set the mask bit corresponding with a cache-line address of the render-cache when the pixel data stored at the cache-line address is read by the graphics engine and the cache-line address is received by the render-cache controller via the pipeline interface, the set mask bit indicating that the pixel data read from the cache-line address is in flight (Hussain, [0043] thru [0050] notes bits used to indicate status of pixel data at a cache line address; Chrysos, column 14, line 53 thru column 15, line 3 notes a sample bit used to indicate a in-flight instruction, column 11, lines 64-67 notes memory).

Art Unit: 2628

As to claim 16, Hussain modified with Chrysos et al. disclose the render-cache controller wherein the pixel mask array indicates whether cache-line addresses included in the thread are associated with the pixel data in flight (column 14, line 53 thru column 15, line 3, Chrysos).

As to claim 17, Hussain modified with Chrysos et al. disclose the render-cache controller wherein the pixel data is in flight if it has been read by the graphics engine more recently than it has been written to the render-cache (Chrysos, column 14, line 53 thru column 15, line 3).

It would have been obvious for a pixel to be read from memory more recently than written to memory because the pixel has to be written to memory first and then read out by the graphics engine to be processed and executed which makes the pixel still in flight until it executed or retired.

As to claim 18, Hussain discloses a method to pre-allocate pixel data to a render-cache, the method comprising:

checking a content addressable memory (tag compare unit) to determine whether pixel data for a particular pixel has been previously allocated to the render-cache (Figure 6, 610; [0043]);

if the pixel data for the particular pixel has not been previously allocated to the render-cache then checking the cache-line status array (cache) to determine an

Art Unit: 2628

address of an available cache-line in the render-cache (Figure 6, 620), evicting pixel data from the address of the available cache-line, and writing the pixel data to the address of the available cache-line in the render-cache (Figure 6; [0044] thru [0050]).

Hussain differs from the invention defined in claim 18 in that Hussain does not disclose setting a bit in a pixel mask array to indicate that the pixel data written to the address of the available cache-line is in flight, the pixel mask array having a plurality of bits, one bit for every entry of the content addressable memory, each bit to indicate whether previously allocated pixel data is in flight; and if the pixel data for the particular pixel has been previously allocated to the render-cache then checking a pixel mask array to determine whether the previously allocated pixel data is in flight, blocking a thread corresponding to the previously allocated pixel data from dispatching to a graphics engine, if the previously allocated pixel data is in flight, and dispatching a thread to the graphics engine if the previously allocated pixel data is not in-flight.

Chrysos et al. disclose setting a bit in a data array to indicate that data written to the address of the available cache-line is in flight, the data array having a plurality of bits, one bit for every entry of a content addressable memory, each bit to indicate whether previously allocated data is in flight; and if the data for

Art Unit: 2628

the particular thread has been previously allocated to the render-cache then checking a data array to determine whether the previously allocated data is in flight, blocking a thread corresponding to the previously allocated pixel data from dispatching to a graphics engine, if the previously allocated data is in flight and dispatching a thread to the graphics engine if the previously allocated data is not in-flight (column 14, line 53 thru column 15, line 3 notes a sample bit used to indicate a in-flight instruction and if the bit is asserted then the data is in-flight and another thread will not be dispatched until that data is retired or aborted; column 26, line 54 thru column 27, line 20).

Chrysos et al. do not explicitly teach pixel data. However, Hussain discloses the pixel data and is used in view of Chrysos et al. system.

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hussain's render cache controller with Chrysos et al's indicating pixel data is in flight to improve the efficiency of the system by rendering data in the order that it should be executed.

As to claim 19, Hussain modified with Chrysos et al. disclose the method wherein checking the content addressable memory to determine whether pixel data for a particular pixel has been previously allocated to the render-cache includes comparing the X and Y coordinates of the particular pixel to X and Y

Art Unit: 2628

coordinates of pixel data stored in the content addressable memory and determining that the pixel data has been previously allocated if the comparison results in a match (Hussain, [0043]).

As to claim 20, Hussain modified with Chrysos et al. disclose the method wherein checking a cache-line status array (cache) to determine an address of an available cache-line in the render-cache includes selecting an available cache-line based on a cache-line selection algorithm (Hussain, [0044]).

As to claim 22, Hussain modified with Chrysos et al. disclose the method wherein evicting pixel data from the address of the available cache-line includes writing the pixel data to a memory (Hussain, [0048] and [0049]).

6. Claim 14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hussain (US 2004/0233208) in view of Chrysos et al. (US 6,549,930) as applied to claim 12 above, and further in view of Baylor et al. (US 2002/0078124).

As to claim 14, Hussain modified with Chrysos et al. disclose the render-cache controller wherein the pixel mask array is to set a bit corresponding with a cache-line address of the render-cache when the pixel data stored at the cache-

Art Unit: 2628

line address is read by the graphics engine and the cache-line address is received by the render-cache controller via the pipeline interface ([0046] thru [0048], Hussain), the set bit indicating that the pixel data read from the cache-line address is in flight (Chrysos, column 14, line 53 thru column 15, line 3), but do not expressly disclose resetting a bit indicating that the pixel data written to the cache-line address.

Baylor discloses resetting a bit indicating that the pixel data written to the cache-line address [0042].

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hussain modified with Chrysos et al.'s method of rendering cache by using bits as a way to allocate data that is read and written to cache which maintains coherency of the memory for future processing.

7. Claim 21 is rejected under 35 U.S.C. 103(a) as being unpatentable over Hussain (US 2004/0233208) in view of Chrysos et al. (US 6,549,930) as applied to claim 20 above, and further in view of Andrews et al. (US 2005/0122339).

Art Unit: 2628

As to claim 21, Hussain modified with Chrysos et al. disclose the method wherein checking a cache-line status array (cache) to determine an address of an available cache-line in the render-cache includes selecting an available cache-line based on a cache-line selection algorithm (Hussain, [0044]) but do not disclose the method wherein the cache-line selection algorithm is based on a least recently used selection algorithm.

Andrews et al. disclose the method wherein the cache-line selection algorithm is based on a least recently used selection algorithm [0067].

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Hussain modified with Chrysos et al's system with Andrew et al.'s selection algorithm to keep accesses to memory for cache line that are used for multiple rendering at a minimum for faster processing and execution.

8. Claims 23, 24 and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over Andrews et al. (US 2005/0122339) in view of Rosenbluth et al. (US 2003/0046488) and Hussain (US 2004/0233208).

As to claim 23, Andrews et al. disclose a portable media device comprising: a main memory (Figure 1, 130); a graphics processor (Figure 1, 102), the

Art Unit: 2628

graphics processor comprising a multithreading, multi-core graphics engine to generate graphics by processing pixel data (Figure 1, 102: note CPU1, CPU2, ...CPU_n denotes multi-core; Figure 2 shows threads), a render-cache to store pixel data (Figure 1, shared L2, 122 accessible to processor 102; [0042]), maintaining data coherency between the render-cache and a main memory [0050], wherein the graphics processor further comprises a raster logic (data generating logic) to generate threads, each thread including at least one cache-line address indicating the location of pixel data in the render-cache (Andrews, Figure 3, 302, 304; [0059], lines 1-18; [0064] notes that the data address corresponding to the cache-line address), and a thread dispatcher to dispatch each thread to the graphics engine only when the render-cache controller indicates that the at least one cache-line address is valid (Andrews, Figure 4 notes validity; Figure 5 notes dispatching threads; [0064]).

Andrews et al. disclose the threads comprises series of subtasks performed in a specific order forming a sequence of such subtasks and the tasks must be executed in the order for rendering scenes [0054].

However, Andrews et al. do not expressly disclose maintaining the order in which each thread is dispatched to the graphics engine in line with the multithreading, multi-core graphics engine processing the pixel data corresponding to each thread and a pixel mask array to identify in flight pixel data.

Andrews et al. do not disclose a battery to provide power to the CPU, the main memory, and the graphics processor. However, it would have been obvious that a system need some source of power supply in order for the components of the system to function at all.

Andrews et al. disclose cache-lines having different fields to indicate certain information about the cache and the data [0064]. However, Andrews et al. also do not disclose the render cache-controller having a pixel mask array to identify in flight pixel data, the render-cache controller having a cache-line status array to identify availability of a cache line in the render cache; and dispatching each thread to the graphics engine only when the render-cache controller indicates a cache hit during a look-up operation and the pixel data stored at the at least one cache-line address is not in-flight.

Rosenbluth et al. disclose a controller to maintain the order in which each thread is dispatched and executed in line with the multi-threading, multi-core processing the data corresponding to each thread and identifying in-flight data ([0074], [0075], [0083] thru [0086] notes that the threads are executed in a strict sequential order and a thread is not executed until the previous is complete; CAM is used to maintain coherency and ensure correct sequential

Art Unit: 2628

processing) and pixel data stored in memory is not in flight ([0084] thru [0086] notes look-up operation and results of cache hit/miss and "in-flight" status).

Therefore, it would be obvious to one of ordinary skill in the art at the time of the invention to modify Andrews et al. system with Rosenbluth's method of maintaining order of threads to ensure the execution of threads in a proper order which will accurately render scenes to be displayed.

Hussain discloses the render-cache controller having a cache-line status array to identify availability of a cache line in the render cache (tag compare unit, [0043]) and dispatching each thread to the graphics engine only when the render-cache controller indicates a cache hit during a look-up operation ([0043] note that the tag comparison of a pixel in memory with a current pixel is interpreted as a "lookup operation").

It would have been obvious to one of ordinary skill in the art at the time of the invention to modify Andrews et al's cache system with Hussain's cache system to maintain coherency of the memory for future processing.

As to claim 24, Andrews et al. modified with Rosenbluth et al. and Hussain disclose the portable media player comprising:
a liquid crystal display to display the graphics generated by the graphics

Art Unit: 2628

processor (Hussain, [0004]).

As to claim 27, Andrews et al. modified with Rosenbluth et al. and Hussain disclose the portable media player wherein the graphics engine is to generate 3D graphics (Andrews, [0033]).

Response to Arguments

9. Applicant's arguments filed May 4, 2009 have been fully considered but they are not persuasive. Applicants argue on pages 8 and 9 of the amendment filed regarding independent claims 1, 11, 18 and 23 that the prior art cited does not teach all of the limitations of the claims. Applicants specifically argue that the prior art does not disclose *dispatching* of threads, but, rather the *execution* of threads. Applicants also argue that because the prior art does not disclose thread dispatching, the prior art could not possibly disclose *blocking* a thread from being dispatched. Per claim 1, the Examiner relies on Rosenbluth to teach the limitations of maintaining the order in which threads are dispatched and blocking a thread from being dispatched. The primary reference, Andrews, refers to a thread as a task performed by a processing unit, typically comprising a series of subtasks performed in a specific order forming a sequence of such subtasks. Rosenbluth teaches signaling for threads to begin within a pipestage [0082]. A thread does not begin until this signal is

Art Unit: 2628

received. This is to maintaining strict sequential ordering of the execution of the threads in a given stage [0083]. Rosenbluth may not explicitly use the term “dispatch,” however, it is obvious that the threads are dispatched in a sense a thread is not provided for execution until signaled.

10. Applicants amend independent claim 23 to recite the limitations of canceled claims 25 and 26. Claim 23 also include limitations of dispatching threads. Rosenbluth is also relied upon to teach the limitations of claim 23 as in claim 1. Please see response above regarding claim 1.

11. Applicants amend independent claim 11 to recite the limitations of canceled claim 15 and also amend independent claim 18 to recite similar limitations of independent claim 1, e.g. “blocking threads from dispatching.” Chrysos is relied upon to teach the limitations of claims 11 and 15 regarding blocking a thread from being dispatched. Chrysos disclose threads or instructions being in-flight from the time it is dispatched until retires or aborts. A bit is used to indicate an “in-flight” thread and remains asserted until the thread aborts or retires (column 14, line 53 thru column 15, line 3). This hinders or “blocks” other instructions or threads from being dispatched and executed.

Conclusion

12. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to JACINTA CRAWFORD whose telephone number is (571)270-1539. The examiner can normally be reached on M-F 8:00a.m. - 5:00p.m. EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kee Tung can be reached on (571) 272-7794. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Jacinta Crawford/
Examiner, Art Unit 2628

/Kee M Tung/
Supervisory Patent Examiner,
Art Unit 2628